Problem 1. Maximum sum with swaps

Input file:	input.txt
Output file:	output.txt
Time limit:	6 seconds
Memory limit:	512 megabytes

Given an array of integers A_i of length N, you are to find an interval within the array, which has maximum sum of the elements in it. However, before choosing the interval, you are allowed to perform no more than K swaps. Each swap makes two elements of the array exchange places.

An interval within the array is an arbitrary set of its **consecutive** elements.

Input

The first line of the input file contains two integers: N — the number of elements in the array ($1 \le N \le 100\,000$), K — the maximum possible number of swaps ($0 \le K \le 10$). The second line lists the elements of the array A_i, one by one. All A_i are integers, no larger than 10^9 by absolute value. It is guaranteed that there is at least one positive number among A_i.

Output

The first line of the output file must contain two integers: S — the resulting sum of elements in the interval and M — the number of swaps performed ($0 \le M \le K$).

The following M lines must describe all swaps in order of their execution. For every j-th swap print two integers u_j and v_j, denoting two positions in the array, whose values are to be swapped ($1 \le u_j \ne v_j \le N$). The positions in the array are numbered consecutively starting from one.

The last line must contain the interval where the sum must be calculated, described as two integers: L being the index of the leftmost position in the array belonging to the interval, and R being the index of the rightmost position belonging to the interval ($1 \le L \le R \le N$).

Examples

input.txt	output.txt
3 2 1 2 3	6 0 1 3
3 3 1 -2 3	4 2 1 2 2 3 2 3
3 0 1 -2 3	3 0 3 3

Problem 2. Inspection

Input file:	input.txt
Output file:	output.txt
Time limit:	1 second
Memory limit:	256 megabytes

A government inspector is coming to your county to check the quality of roads. You want to pass the inspection with minimal effort and expenses, so you're planning to take the inspector on a planned round trip between towns, in order to avoid repairing roads all over the county. Any road in the county goes straight from one town to another, and all roads are one-way. To drive the inspector around successfully, you may have to build new roads. Note that roads going out of a town and back again to the same town are forbidden, and building them even to fool state inspectors is considered bad taste. However, you can build as many roads between any two towns as you want, with arbitrary directions. Define the minimal number of roads to be built in order to take the inspector on a round trip. The length of the trip **does not** matter.

Input

The first line of the input file contains two integers: N — the number of towns and M — the number of one-way roads in the county ($1 \le N \le 10^5$, $0 \le M \le 10^5$).

The remaining M lines of the file contain the descriptions of the available roads in the county. For each road, a separate line contains two integers: A — the number of the town where the road begins and B — the number of the town where the road ends ($1 \le A \ne B \le N$). All towns are numbered with integers from 1 to N.

Output

The output file must contain a single integer — the minimal number of roads that must be built in order to be able to take the inspector on a round trip between the towns. If that is impossible, print -1.

Example

input.txt	output.txt
2 5 2 1 2 1 2 1 2 1 1 2	0
3 2 2 3 1 2	1

Problem 3. Karmon go

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 megabytes

As part of the import phaseout government plan in you-know-what country, a game called "Karmon go" was released; the game mechanics is based around karaoke monsters, the Karmons. Among other things, each karmon in the game has a battle power parameter — the BP. BP can be increased by two means: by *powering up* the karmon and by *evolving* it into another karmon species. When a karmon is powered up, a certain constant is added to its BP, which depends only on the species of the karmon. When a karmon evolves, BP is multiplied by a certain factor, depending only on the species of the karmon into which it evolves. A karmon can be evolved only once, and can be powered up only a limited number of times. The player must use both methods of increasing his karmon's BP, but he's free to choose the order: to power up and then to evolve, or vice versa. The resulting BP depends on the chosen order.

Consider the following example: let a player have a Bichu karmon with a BP of 7. Powering up Bichu raises its BP by 5. Bichu can be evolved into Bikachu, multiplying its BP by 1.6. Powering up Bikachu raises its BP by 7. Let a player, for example, power his karmon up 3 times. First powering Bichu up and then evolving it results in a BP of $(7 + 3 \cdot 5) \cdot 1.6 = 35.2$. On the other hand, first evolving Bichu into Bikachu and subsequently powering it up brings its BP to $7 \cdot 1.6 + 3 \cdot 7 = 32.2$. It turns out that in the current example powering up followed by evolving is more advantageous.

Help our rookie players figure out the best strategy to develop their karmons: powering up then evolving or vice versa.

Input

The input file consists of a single line describing a karmon's features and its evolution in the following format: $name_0$ inc_0 $name_1$ mul_1 inc_1,

- $name_0$ — karmon name.
- inc_0 — integer constant added to the karmon's BP when powered up prior to evolution ($1 \le inc_0 \le 10^6$).
- $name_1$ — karmon name after evolution.
- mul_1 — real number factor, by which the karmon's BP is multiplied after evolution ($1 \le mul_1 \le 100$). The number contains no more than five decimal digits after decimal point.
- inc_1 — integer constant added to the karmon's BP when powered up after evolution ($1 \le inc_1 \le 10^6$).

The name of any karmon can contain only Latin characters in upper and lower case. The length of the name should be no less than one and no longer than a hundred characters.

Output

The output file must contain the line "Power up, Evolve" or the line "Evolve, Power up" depending on which way of developing the karmon is more advantageous to achieve the highest BP: first power up and then evolve, or the opposite, respectively. If the order of powering up and evolution makes no difference, print the word "Whatever".

Example

input.txt	output.txt
Geevee 10 Gaporeon 2 20	Whatever
Geevee 10 Golteon 1.2 11	Power up, Evolve
Geevee 10 Glareon 1.3 14	Evolve, Power up

Problem 4. Park trails

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	3 seconds
	6 seconds (for Java)
Memory limit:	256 megabytes

The Forest Frontiers theme park was recently inspected by the Federal Emergency Management Agency. The most serious problem revealed was the layout of trails in the park. Help the park management fix this problem.

The park map can be easily represented on the coordinate plane. Each trail in the park is a line segment parallel to one of the coordinate axes. Any two trails don't have common points, with one exception: end points of segments may coincide. An evacuation point is located somewhere in the park. This point is guaranteed either to lie off all the trails or to coincide with an end point of a trail.

In case of an emergency, an alarm is switched on, and visitors are told to proceed to the evacuation point. According to the FEMA rules, any visitor must be able to reach the evacuation point moving only along trails, with an additional constraint that the distance to the evacuation point must decrease monotonously during movement. It is assumed that park visitors never leave the trails, so each visitor can be located at arbitrary point of arbitrary trail when the alarm is triggered.

Obviously, no one took this silly regulation into account when planning the park layout.

At the moment, the above-ground part of the park is finished, and it's impossible to build new trails above ground. However, it's possible to build some trails below ground, in tunnels. Underground trails are treated the same way as regular trails in FEMA regulations. The difference in elevation is **not** taken into account when calculating the distance to the evacuation point.

We can safely assume that each tunnel consists of a set of underground trails. Additional FEMA rules apply to tunnels:

1. A tunnel cannot be forked: there are strictly two entrances and a linear path between them.
2. Each tunnel entrance must be located on an above-ground trail or at the evacuation point.

Tunnels can be built at varying depths, so they can intersect on planar park map.

Builders calculate the cost of tunnel construction based on its length on the park map. The vertical component (the depth of the tunnel) is not taken into account. Suggest how to build tunnels in order to comply with the FEMA requirements, so that the total length of all tunnels is minimal.

Input

The first line of the input file contains an integer N — the number of trails in the park ($1 \le N \le 100\,000$). The second line contains two numbers x_e and y_e — the evacuation point coordinates.

Each of the remaining N lines describes a single trail with four numbers x_1, y_1, x_2, y_2, where x_1, y_1 are the coordinates of one end of the trail and x_2, y_2 are the coordinates of the other end. It is guaranteed that $x_1 = x_2$ and $y_1 \ne y_2$ or vice versa.

All coordinates are integer. Absolute value of any coordinate does not exceed 10^8.

Output

The first line of the output file must contain two integers: K — the number of tunnels to be built and A — the total length of these tunnels. Each of the following K lines must contain a description of a single tunnel.

Each tunnel can be represented as a polyline with segments parallel to the coordinate axes. The description of a tunnel must begin with the number of vertices in the polyline (at least two), followed by x and y coordinates of the polyline vertices in order (from one end to another). All numbers must be integers. Any two adjacent vertices of the polyline must have one coordinate equal, and the other one different.

The total number of vertices in all tunnels you're planning to build must not exceed 10^6. It is guaranteed that there exists an optimal plan complying with the limitations of the output format. If several solutions are possible, print any of them. If the park already complies with the FEMA rules as is, and no tunnels are necessary, the output file must contain two numbers $K = A = 0$.

Example

input.txt	output.txt
12 0 0 1 3 1 5 3 1 5 1 1 3 2 3 2 3 2 2 2 2 3 2 3 1 3 2 5 1 5 3 5 3 4 3 4 3 4 4 4 4 3 4 3 4 3 5 3 5 1 5	5 16 3 0 0 1 0 1 3 3 2 2 0 2 0 0 4 0 0 1 0 1 1 3 1 3 2 3 3 3 3 4 3 3 2 3 3 4 3

Problem 5. Autocomplete

Input file:	input.txt
Output file:	output.txt
Time limit:	1 second
Memory limit:	256 megabytes

Two words are considered *similar*, if they are equal when compared in case-insensitive way, but in a case-sensitive comparison they differ in no more than K positions.

A dictionary containing W words as well as Q query-words is given. For each query-word, print a single integer: the number of similar words in the dictionary.

Input

The first line of the input file contains an integer K — the maximum number of positions at which the words can differ by case ($0 \le K \le 5$).

The second line contains an integer W — the number of words in the dictionary ($1 \le W \le 1\,000$).

The following W lines contain the dictionary, one word per line. Each line consists of small and capital Latin letters. All words are non-empty and are no longer than 2 000 symbols.

The following line contains an integer Q — the number of queries ($1 \le Q \le 1\,000$).

The next Q lines contain queries, one word per line. Same as with the words in the dictionary, each query consists of capital and small Latin letters, all queries are non-empty and no longer than 2 000 symbols each.

Output

For each of Q queries from the input file print a single integer: the number of similar words in the dictionary. Answers to the queries must be printed in the same order as the queries are listed in the input.

Example

input.txt	output.txt
2 5 theword TheWord THEWORD thewordandsomeletters theword 4 theword The theword TheWordAndSomeLetters	3 0 3 0

Problem 6. Amazing Divisibility

Input file:	input.txt
Output file:	output.txt
Time limit:	1 second
Memory limit:	256 megabytes

The sun was setting. Teddy sat in the bus 127, stuck in traffic. Bored, he'd already figured out how to get a hundred by juggling with the digits of his ticket number. He noticed that if he added the digits of the number of the bus to those of the year of his birth, 1997, he got a number divisible by 7. «Heh!», — Teddy thought and wrote 12 after the bus number, because that day was the 12th. Again, he got a number divisible by 7. The evening was becoming fun. Finally, he wrote the day of the month after the year of his birth.

"Coincidence? I don't think so!", — Teddy wondered and dashed out of the bus to get home as fast as he could run, to write a program to evaluate the probability of his results.

Help Teddy write a program that finds the number of ways to choose a pair of numbers from a given set, such that if the digits of the second number are added to those of the first, the resulting number is divisible by 7.

Input

The first line of the input file contains an integer N — the number of numbers($2 \le N \le 10^5$).

The second line contains N space-separated different integersa_i ($1 \le i \le N$, $1 \le a_i \le 10^9$).

Output

The output file must contain a single number - the number of ways to choose a pair of numbers from a given set, such that if the digits of the second number are added to those of the first, the resulting number is divisible by 7.

Example

input.txt	output.txt
3 127 1996 12	4
4 11 2 1 12	4

Example explanation

In the first test a total of 6 pairs can be made of the three numbers. If the number 1996 is written after 12, the resulting number is $121996 = 7 \cdot 17428$. The three remaining pairs are described in the problem body.

Problem 7. Group tournament

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	2 seconds
	4 seconds (for Java)
Memory limit:	256 megabytes

In our capitalist-run, dog-eat-dog world money is everything, and big sports are no exception. All participating teams have already bought enough points for the next season, and all the local hockey federation has to do now is distribute the results of the upcoming games. However, some teams felt generous and apart from buying points also bought the results of some matches. Initially the federation officials thought it would only make their life easier: the more games are fixed, the less work. It was only later that they understood their wrong and asked us to be a part of their scheme and help them distribute the results of the games in the upcoming season.

The local hockey tournament follows a round scheme: N teams participate, each team plays a game against each and every other team strictly once. Teams score points for games according to the following rules:

- If the winning team is defined at the end of the regulation time of the match, it scores 3 points, and the other team gets none.
- If a game is tied after regulation time, overtime ensues. In this case the winner gets 2 points and the loser gets 1. The overtime is unlimited and lasts until someone scores a goal.

Based on the tournament results, a team's score is calculated as the sum of its points earned in all games played.

Input

The first line of the input file contains an integer N — the number of tournament participants ($2 \le N \le 100$). Teams are numbered from 1 to N.

The following N lines of the file each contain N symbols and are in essence a tournament table for the given moment of time.

The symbol a_{ij} in the line i and position j denotes the result of a game scheduled to be played by team number i against team number j ($1 \le i, j \le N$). It can be one of the following:

- `'W'` — means that the team i will win a match against j in the regulation time
- `'w'` — the team i will win in a match against j in the overtime
- `'l'` — the team i will lose to the team j in the overtime
- `'L'` — the team i will lose to the team j in the regulation time
- `'.'` — if the result of the game between i and j is not yet determined
- `'#'` — if i equals j, it means that there is no such game, i.e. a team cannot play against itself.

It is guaranteed that the table is correct. More formally:

- a_{ij} = `'#'` for all $i = j$
- if a_{ij} = `'.'`, then a_{ji} = `'.'`
- a_{ij} = `'W'` when and only when a_{ji} = `'L'`
- a_{ij} = `'w'` when and only when a_{ji} = `'l'`

The last line of the input file contains N integers p_i — the number of points the i-th team must score ($1 \le i \le N$).

Output

The output file must contain a completely filled tournament table in the same format as that in the input file.

It is guaranteed that a solution exists. If there are several solutions, print any of them.

Example

input.txt	output.txt
4 #..W .#w. .l#. L..# 8 6 3 1	#wWW l#wW Ll#w LLl#

Problem 8. Multiplier

Input file:	input.txt
Output file:	output.txt
Time limit:	1 second
Memory limit:	256 megabytes

In digital circuit engineering, computational algorithms are implemented using operation blocks. An operation block has one or several inputs and a single output. Various types of blocks perform various mathematical operations. Values on inputs and outputs are integers. A value on the output is defined by the type of the block and its input values.

Often, a circuit for a complex operation can be constructed from blocks of simpler operations. In this case, the input of each block must receive values either from another block output or from one of the circuit inputs. Circular dependencies in the circuit are forbidden: value on a block input cannot depend on the value on the block output, neither directly nor indirectly. Value from any block output may be transmitted to any number of inputs of other blocks. The output value of one of the blocks in the circuit is designated to be the output value of the whole circuit.

In the current problem, build a circuit of a multiplier by a defined number N. The circuit has strictly one input, which receives the value x. The resulting output value of the circuit must be $N \cdot x$, i.e. the product of N multiplied by the input value x. The following block types are allowed:

1. Addition block: has two inputs. The output of this block produces the sum of values received at its first and second inputs.

2. Subtraction block: has two inputs. The output of this block produces the difference of the value received at the first input and the value received at the second input.

3. k-shift block: has one input. The output of this block produces the value received at its input multiplied by 2^k.

In the current problem it is *forbidden* for the shift block to receive its input value from the addition or subtraction blocks. Find out the minimal number of blocks required to produce the necessary N-multiplier circuit.

Input

The only line of the input file contains a single integer N — the multiplier parameter ($2 \le N \le 10^3$).

Output

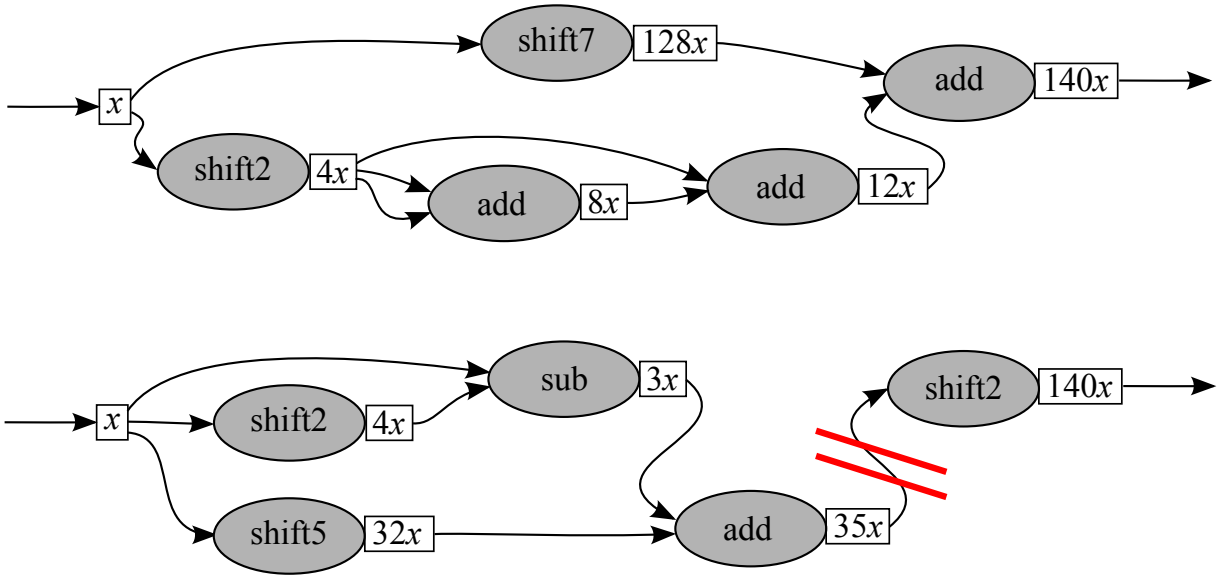
The output file must contain a single integer — the minimal number of operation blocks needed to construct the multiplier.

Example

input.txt	output.txt
4	1
140	5

Example explanation

Provided below are two examples of circuits of a multiplier by 140 made up of 5 blocks. The circuit at the bottom is *forbidden by the problem specification*. Shift blocks, addition blocks and subtraction blocks are denoted by shift, add and sub, respectively.


x
shift7
128x
add
140x
shift2
4x
add
8x
add
12x

x
shift2
4x
sub
3x
shift5
32x
add
35x
shift2
140x

Problem 9. Guess the modulo

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 megabytes

Fimka Sobak, unlike Ellochka the Cannibal, was reputed to be a cultured girl. In addition to her favorite smart buzzword, the . . .-ality, she learned a remarkably entertaining game, "Guess the modulo".

The game is played in the following way. There are two players: the leader and the player. The leader initially thinks of two numbers: N — the number of numbers to be summed and M — the modulo. The leader tells the player the number N, and the player must guess the number M. Moreover, the leader thinks of an array of $(N-1)$ numbers and tells it to the player.

During the game the player tells the leader numbers to get more information. Every time the player says a number, the leader:

1. appends it to the end of his array,
2. calculate S — the sum of the last N numbers in the array by the modulo M,
3. appends S to the end of the array and
4. tells S to the player.

As soon as the player guesses the value of the modulo M, he or she informs the leader.

It is known that the modulo M lies within the range between 2 and 10^9, inclusively.

Interaction Protocol

This problem is interactive, and instead of file input-output you'll have to work with a special program — an interactor. Interacting with the program is performed through the standard input-output streams.

First you will be given the number N and the first $(N-1)$ numbers of the array. Next, your programs will be sending queries. Each query for an addition of a number to the array will retrieve a new sum modulo according to the game rules. After the query guessing the modulo the player's solution must terminate.

If you can't guess the modulo or guess wrong, you will get the verdict Wrong Answer.

The number of player's queries, including the final modulo-guessing query, must not be greater than 10^3. If you exceed the limit of the number of queries, you will get the verdict Wrong Answer.

Input

The first line of the input stream contains a single integer N ($2 \le N \le 100$). The following line contains $(N-1)$ space-separated integers ranging from 0 to 10^9 inclusively.

The following lines of the input stream contain answers to questions. Each answer is the sum of the last N elements of its array by modulo M.

Output

You should print the queries and the player's answer into the standard output.

Player query format: "`? p`", `p` — the number added by the player to the array. The number p must be integer and must lie within the range from 0 to 10^9 inclusively.

Player answer format: "`! M`", `M` — the modulo in question.

Make sure that each query ends with a line break and that you flush the output stream buffer (the `flush` command of the language). Otherwise the solution may get the verdict Deadlock.

Example

standard input	standard output
4 1 2 3 6 0 6	? 0 ? 5 ? 2 ! 7

Problem 10. Carts

Input file:	input.txt
Output file:	output.txt
Time limit:	1 second
Memory limit:	256 megabytes

Peter decided to prepare for the new academic year seriously: he dare to buy a new smartphone. After reading reviews, he made up his mind and chose the Psiomi Ni 42 model. To earn the money for the new gadget, Peter took a job in a local mall. After the interview, he was hired to remove carts from the parking lot. The essence of the job was the following — he was supposed to collect empty shopping carts together from the mall parking lot and roll them back to the mall. Obviously, Peter doesn't like wasting his time and efforts, so he came up with an idea how to make the job as easy as possible. You may have guessed that Peter is already hard at work, reading forums on how to root his new phone. Meanwhile, try to solve the task that Peter has already solved on his mall job.

To make things simpler, we may assume that the parking lot is a plane with Cartesian coordinates, infinite in any direction. Each cart takes a single square cell with integer vertex coordinates. Let's assume that in a unit of time, a single cart can be moved left, right, up or down the plane to the adjacent integer square, provided that it's not occupied by another cart.

Find the minimal number of cart moves required in order to place the carts into a line of adjacent cells parallel to one of the coordinate axes.

Input

The first line of the input file contains a single integer n — the number of carts in the lot ($2 \le n \le 10^5$).

The following n lines describe the positions of the carts. Each cart position is described with two space-separated integers x and y — the coordinates of the bottom left corner of the corresponding square on the plane ($0 \le x, y \le 10^9$).

It is guaranteed that each cell contains no more than one cart.

Output

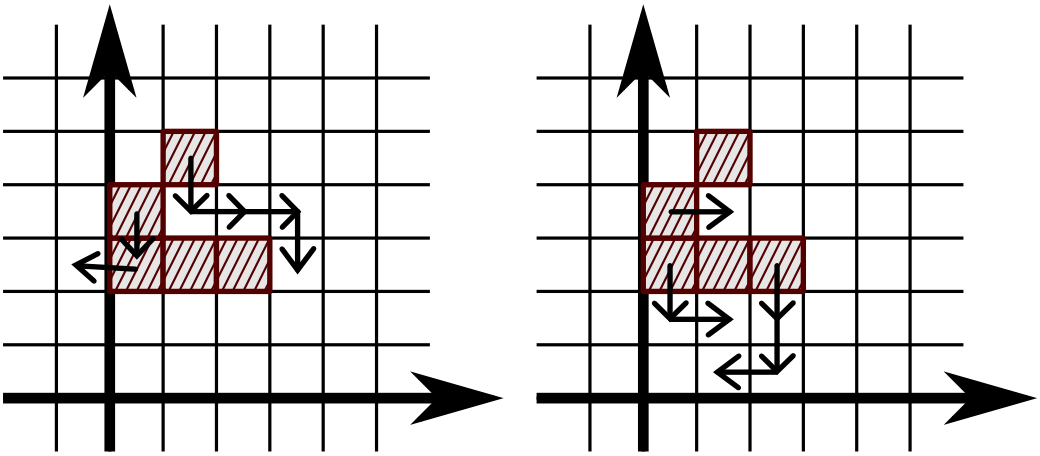
The output file must contain a single integer — the minimal number of cart moves.

Example

input.txt	output.txt
5 0 2 1 2 2 2 0 3 1 4	6

Example explanation

It is possible to place the carts parallel to any of the axes using six moves in the example:



Problem 11. School informatics

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	3 seconds
Memory limit:	256 megabytes

Vasya was getting ready for his SAT tests in Informatics and found the following problem in one of his textbooks: "All books stored in the library have the same format. A book contains 400 pages, each page contains 40 lines, and there are precisely 80 printed symbols in a line. There are 25 different symbols: 22 letters, period, comma and space. Calculate the number of bits necessary to encode the contents of a single book."

Vasya decided that the authors of the problem assumed that all symbols must be coded with the same number of bits. Thus, to code 25 different symbols one must use 5 bits per symbols. But such coding means storing unnecessary information.

Had it been known that different symbols and their combinations occur in text with different frequency, using variable-length code would make sense, e.g. Huffman code. But the authors hadn't provided the necessary information, so Vasya assumed all symbols and their combinations occur in the text with equal frequency, and used a fixed number of bits for coding.

Vasya gave it another thought and realized that coding groups of symbols instead of single symbols could allow to partially get rid of the useless information.

For example, when coding groups of three subsequent symbols, the number of possible combinations would be $25^3 = 15625$. Then 14 bits is enough to encode this number of combinations. This way we get $4\frac{2}{3}$ bits per symbol instead of 5 as the authors assumed.

Vasya wondered if this was a way to indefinitely approach the value $\log_2 25 \approx 4.64385619$. But then he remembered that the code was intended for use with finite-length messages. If the length of a message is not divisible by the length of the letter groups being coded, the message is automatically padded with spaces until number of letter groups becomes integer. Moreover, the use of excessively long letter groups is impossible due to technical reasons.

Vasya decided to write a program which would define the optimal size of letter group for coding messages with predefined parameters. Help Vasya write the program.

Input

The first line of the input file contains the integer number T — the number of tests ($1 \le T \le 10^4$). This is followed by the test descriptions, one test per line.

For each test, three integer numbers are provided: N — the number of different symbols in the message alphabet, L — the message length in symbols, and K — the maximum allowed letter group size ($2 \le N \le 10^9$, $1 \le L \le 2 \cdot 10^6$, $1 \le K \le L$).

For each test it is guaranteed that the number $\log_2 N$ either is an integer, or differs from any rational number with a denominator no greater than K by at least 10^{-13}.

Output

For each test, print a separate line containing two integers: A — the resulting length of the coded message in bits and B — the size of the letter group used ($1 \le B \le K$).

The length of the coded message A must be as short as possible, provided that the length of the letter group B is no greater than the number K (defined in the input data). If there are several optimal solutions, any of them can be printed.

Example

input.txt	output.txt
3 25 1280000 10 2 1000 1000 3 7 3	5973338 3 1000 1000 14 1

Example explanation

The first test corresponds to the example from Vasya's textbook: the message is 1 280 000 symbols long, and the alphabet contains 25 symbols. In the given case, the use of three-letter groups provides the minimal size of the encoded message, if letter group size is kept within 10. One space is automatically added at the end of the message due to padding. If each symbol was coded separately, as originally intended by the textbook authors, 6 400 000 bits of information would be required.

Problem 12. Give the Parabellum away

Input file:	input.txt
Output file:	output.txt
Time limit:	2 seconds
	3 seconds (for Java)
Memory limit:	256 megabytes

Ostap is strolling leisurely along the Yessentuki-Moscow route, and scuttling along him is Kislarsky, begging to take his Parabellum away. Kislarsky keeps distance to Ostap strictly constant. Also he keeps constant his speed relative to the ground. He is moving counterclockwise relative to Ostap. Ostap's velocity stays constant (both direction and magnitude-wise).

Help Kislarsky to get away from the Alliance of the Sword and Ploughshare. Find the coordinates of the points where he will be in the given time moments t_i.

Input

The first line of the input file contains eight integers: p_x, p_y, q_x, q_y, u_x, u_y, v, N, where:

- p_x, p_y — the location of Ostap in the initial moment ($|p_x|, |p_y| \le 10^4$),
- q_x, q_y — the location of Kislarsky in the initial moment ($|q_x|, |q_y| \le 10^4$),
- u_x, u_y — the projections of Ostap's velocity upon the coordinate axes OX and OY respectively ($|u_x|, |u_y| \le 10$),
- v — the speed of Kislarsky relative to the ground ($\sqrt{u_x^2 + u_y^2} + \frac{1}{2} < v \le 10$),
- N — the number of time moments when the location of Kislarsky is of interest ($1 \le N \le 100\,000$).

The second line contains N real numbers t_i – the moments in time for which the location of Kislarsky must be found ($0 \le t_i \le 1000$). All numbers t_i are provided with at most five digits after decimal point.

It is guaranteed that initial locations of Ostap and Kislarsky are different.

Output

The output file must contain N pairs of real numbers (two per line): X and Y coordinates of Kislarsky's location at the time moment t_i.

The absolute or relative error of each number must not exceed 10^{-5}.

Example

input.txt	output.txt
2 3 4 5 1 1 2 10 1 2 3 4 5 6 7 8 9 10.00	3.311667781 6.811203158 1.618058050 6.525238521 2.396313856 4.895093459 4.356603786 4.697990251 6.268584256 5.267779107 8.050839947 6.172029827 9.724368272 7.265902232 11.304988035 8.490616365 12.800188718 9.818413289 14.210788762 11.235796766